VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

EKONOMICKÁ FAKULTA

BAKALÁŘSKÁ PRÁCE

2010 **Josef Koňařík**

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Evidenční databáze s fakturací

Registration database with invoicing

Student : **Josef Koňařík**

Vedoucí bakalářské práce : **RNDr. Antonín Prcín**

Valašské Meziříčí 2010

Prohlašuji, že

- jsem byl seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;

- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou (bakalářskou) práci užít (§ 35 odst. 3);

- souhlasím s tím, že bakalářská práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že bibliografické údaje o bakalářské práci budou zveřejněny v informačním systému VŠB-TUO;

- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;

- bylo sjednáno, že užít své dílo, bakalářskou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne …….

……
jméno a příjmení studenta

Adresa trvalého pobytu studenta:
……

Poděkování :

Děkuji panu RNDr. Antonínu Prcínovi za odbornou pomoc a čas strávený při vedení mé bakalářské práce.

Obsah

1. Úvod

Výpočetní techniku využívá v dnešní době každá společnost. V některých případech se bez ní vůbec nelze obejít a je nezbytnou součástí daného podnikání. Informační systémy a další programové vybavení (software) usnadňují uživatelům práci s informacemi. Evidence a další možnosti zavedení software v oboru daného podnikání zvedne konkurenceschopnost vůči větším společnostem. Pomocí databází jsme schopni uchovávat data v elektronické podobě a rychleji k nim přistupovat. Tento postup je mnohem jednodušší, než vyplňování papírových formulářů a poté jejich vyhledávání ve skladu s ostatními papírovými daty.

Vzniká už jen problém seznámit s novou technologií zaměstnance, proškolit je nebo přijmout jiné s dostatečnou kvalifikací v oboru IT. S tímto napomáhají školící střediska a školy, které se více zaměřují na novodobé technologie a snaží se vyhovět všem požadavkům trhu práce. V poslední řadě zbývají už jen finance dostačující k rozvoji IT.

Ve své bakalářské práci jsem se rozhodl vypracovat evidenční databázi s fakturací pro auto-pneu servis. V porovnání s jinými servisy nevedou žádné dokumenty o opravě, s čím se setkáváme málokdy. Hlavní zásadou bude zapsání automobilu se základními informacemi do databáze, poté možnost přiřazení faktur k vozidlu s klasickými daty vyskytující se ve všech fakturách, fakturační údaje o majiteli a zákazníkovi, popis závady, postup při odstranění závady, počet vyměněných dílů a v neposlední řadě cena celkové opravy.

Vše budu řešit pomocí aplikace Access, která se nachází v balíčku Microsoft Office, konkrétně v Office 2003. Výhoda této aplikace spočívá v jednoduchosti návrhového zobrazení, kde budu specifikovat většinu úkonů souvisejících s tvorbou tabulek, dotazů a formulářů. K tvorbě událostních procedur využiju programovacího jazyka VBA, jež je přímo integrován v již zmiňované aplikaci.

2. Popis současného stavu

Autoservis Lukáš Burdík, jak už vyplývá z názvu, je společnost vedená na živnostenský list, kde je majitel jako mechanik spolu s dalším odborným pracovníkem. Společnost se zabývá opravou všech značek vozidel, mezi jejich hlavní práce patří například :

- veškeré mechanické opravy
- karosářské práce
- servis klimatizací
- pneuservis
- diagnostika motoru
- příprava vozidel na STK

Její sídlo je ve Vsetíně u fotbalového stadiónu v místní části zvané Ohrada. Společnost zajistí kvalitní náhradní díly z prodejen s kvalitním zbožím, které se nacházejí nedaleko servisu. Také odborně poradí při výběru ojetého nebo nového vozu. Servis také vlastní diagnostický program, ve kterém není problém zjistit stav vozidla a jeho následná oprava.

Od majitele Lukáše Burdíka jsem se dozvěděl základní služby, které společnost nabízí. Jelikož jsem také jejich stálým zákazníkem navrhl jsem vypracovat databázi vozidel, které již opravili pro přehlednější a rychlejší přístup k závadě konkrétního vozidla. Budou také předcházet nákupům nekvalitních, leč na první pohled funkčních dílu do vozidel z poznámek zaznamenaných při montáži a opravě, kdy je také zapsána značka náhradního dílu (pokud se nejedná o originál). Takto se v budoucnu vyvarují nákupem „zmetků“.

2.1 Hardware a software ve firmě

Hardwarové vybavení firmy není příliš velké. Sestava postaršího počítače s malou kapacitou a nedostatečným výkonem pro diagnostický software nevyhovovala, a tak se rozhodl majitel investovat do lepšího vybavení a to

nákupem notebooku značky Acer, dle mého doporučení velmi dostačujícího pro jakoukoliv práci v servisu. Dále firma disponuje multifunkční tiskárnou Canon pro tisk přehledu pracovních úkonů a cen na opravovaných vozidlech. Tento přehled slouží i jako faktura.

Firma vlastní také diagnostický kabel sloužící k propojení počítače s vozidlem vhodný do několika typů vozidel. Pro objednávky méně dostupných náhradních dílu využívají internetové připojení Vsetínského poskytovatele SychrovNET. Dalším softwarem jsou diagnostické programy zasahující pořizovací cenou do stovky tisíc korun a balíček sady Microsoft Office 2003.

3. Teoretická východiska

Databázové technologie se již v polovině minulého století staly neodmyslitelnou součástí technologií informačních. Stejně jako každé další programové vybavení a technologie obecně i databázová problematika prošla bouřlivým vývojem, který stále ještě není u konce. A vlastně ani nikdy být u konce nemůže. Databáze se musí vyvíjet spolu s vývojem dalších základů informačních systémů – hardwarem počínaje a komunikačními technologiemi konče. [4]

Za databázové lze v dnešní době považovat drtivou většinu existujících aplikací – od jednoduchých evidencí přes účetní programy či internetové portály až po podnikové informační systémy. Se službami využívajícími databáze se dnes setkáváme na každém kroku, na poště při odesílání doporučeného dopisu, na letišti při odbavování k letu či kdekoli při zavolání z mobilu se do databáze operátora uloží záznam o uskutečněném hovoru. [4]

Aplikace vhodná na tvorbu databází, ať už se jedná o menší nebo větší projekt, je bezesporu Miscrosoft Access, který je součásti balíčku Office. Není tedy nutné koupě dalších softwarových produktů. V dalších kapitolách si popíšeme základní prvky v již zmiňované aplikaci.

3.1 Tabulky

Návrh tabulky je základním kamenem pro relační databáze. Tento fakt by se neměl jen tak podceňovat a věnovat se mu podrobně. Struktura a obsah tabulek, stejně jako relace mezi jednotlivými tabulkami mohou významně ovlivnit služby, které mohou poskytovat zákaznické databáze svým uživatelům. Samotná struktura tabulek může také ovlivnit rychlost uživatelských řešení, stejně rozšíření systému o další služby.

Každá tabulka v databázi by měla obsahovat pouze informace, jež jsou charakteristické pro určitý typ entity. Například typická škola by měla mít svoje studenty, učitele a vyučovací hodiny. Studenti se zapisují do vyučování, kdežto učitelé vyučování vedou. Databázová aplikace pro potřeby školy by měla obsahovat tabulku studentů, tabulku pro vyučovací hodiny a tabulku učitelů. Tabulka učitele by měla obsahovat informace o učitelích, nikoliv však o studentech nebo vyučování. Podobně by tabulka vyučování neměla obsahovat adresy, čísla pojištění, či telefonní čísla učitelů. Tabulky databází mají podobnou struktury jako listy tabulkových procesorů. Řádky nebo také záznamy v tabulce reprezentují jedinečné instance entit uložených v tabulce. Každé pole (sloupec) v tabulce uchovává charakteristický typ informací. Mnoho databázových tabulek obsahuje jedno nebo několik polí, jež jednoznačně určují každý řádek v tabulce. Této jednoznačné identifikaci se říká primární klíč. [1]

Názvy polí v tabulce mají maximální velikost 64 znaků, a nesmí obsahovat tečky, vykřičníky, hranaté závorky ani apostrofy. Dále nesmí začínat mezerou nebo jiným řídícím znakem (hodnoty ASCII od 0 do 31). [1]

3.1.1 Normalizace

Pojem normalizace zahrnuje uplatnění souborů návrhových pravidel na tabulky databáze. Normalizace skýtá přinejmenším čtyři výhody :

- Odstraňuje přebytečné informace. Mnoho nenormalizovaných databází vyžaduje, aby byla stejná kontaktní informace vkládána na několika různých formulářích. Vyloučením této nadbytečnosti se omezuje pravděpodobnost

výskytu chyb při zápisu, které by mohly poškodit databázi. Normalizace také zjednodušuje správu databáze, protože každá hodnota je uložena pouze na jednom místě, a tudíž pro její úpravu nebo odstranění z databáze postačí jeden krok. [1]

- Zmenšuje velikost databáze. Databáze nemusí ukládat vícenásobné kopie stejné informace, protože každý typ informace je uložen jen na jednom místě. Normalizace také minimalizuje počet sloupců v tabulce, což ve svém důsledku také zmenšuje celkovou velikost databáze. [1]

- Zjednodušuje vyhledávání. Profesionálové, kteří rozumí pravidlům normalizace, budou okamžitě vědět, jak procházet tabulky databáze a vyhledat požadovanou informaci. Příležitostnému uživateli databáze bude logika jejího uspořádání připadat velmi intuitivní, protože každá tabulka popisuje samostatnou entitu a sloupce tabulky jsou vlastnostmi této entity. [1]

- Zjednodušuje dotazování. Sloupec tabulky uchovává jeden typ dat jako jméno nebo příjmení, nikoliv oba najednou. Tím, že uložíte příjmení do samostatného sloupce databáze, může snáze zhotovit seznam všech řádků vztahujících se k určenému příjmení. V nenormalizovaných databázích, v nichž je jméno a příjmení uloženo v jednom sloupci, musí dotaz nejprve odvodit příjmení a teprve pak vyhledat specifické informace týkající se této osoby. [1]

3.1.2 Datové typy

Při tvorbě tabulek je potřeba vybrat správný datový typ, ať už tabulku vytváříme pomocí průvodce nebo ručně. Aplikace Access nabízí deset možností na výběr požadovaného datového typu. Každý typ má svou velikost a podle toho se odvíjí její maximální délka dat v poli. Vyjmenujeme si je zde s menším popisem :

- Text – obsahuje data z alfanumerických znaků, maximální velikost 255 znaků. Tohle pole je možné indexovat, můžeme tedy podle toho pole řadit nebo dělat výběry.

- Memo – stejný typ jako Text, jeho předností je velikost 64 000 znaků, indexovat lez ale pouze prvních 128 znaků.

- Číslo – zde můžeme vkládat numerické typy dat a to o těchto velikostech čísel:

  - Bajt (Byte) – číselná hodnota od 0 do 255, velikost 1bajt,
  - Celé číslo – číselné hodnoty od -32 768 do 32 767,velikost 2 bajty,
  - Dlouhé celé číslo – číselné hodnoty v intervalu od -2 147 483 648 do 2 147 483 648, velikost 4 bajty,
  - Jednoduchá přesnost – číselné hodnoty od - 3,402823E38 do 3,402823E38, přesnost na 7 desetinných míst, 4 bajty s pohyblivou řádovou čárkou.
  - Dvojitá přesnost – číselné hodnoty od - 1,79769313486232E308 do 1,79769313486232E308, přesnost na 15 desetinných míst, 8 bajtů s pohyblivou řádovou čárkou.
  - Replikační identifikátor -16 bajtový globální jednoznačný identifikátor.
  - Desetinné číslo -12 bajtové číslo s předpokládanou desetinnou přesností max. 28 desetinných míst obsahující hodnoty od –1028 do +1028. Je nutné definovat přesnost a měřítko.

- Datum/Čas – pole může obsahovat hodnoty kalendářního data v intervalu 100 až 9 999 a času v intervalu 0:00:00.00 až 23:59:59.59. Jde o zvláštní numerický typ, neboť datum a čas si Access převede na pořadové číslo od 1.1.1900, přičemž datum tvoří celou část čísla a čas část desetinnou. Velikost 8 bajtů.

- Automatické číslo – číslo generované automaticky se používá většinou jako primární klíč, nejde jej aktualizovat ručně, jeho hodnoty jsou ideální pro

jedinečné označení. Můžeme mít narůstání čísla o jedničku nebo náhodné číslo.

- Měna - obsahuje hodnoty měny zahrnující data s jedním až čtyřmi desetinnými místy s přesností na 15 číslic vlevo od desetinné čárky. Velikost 8 bajtů.

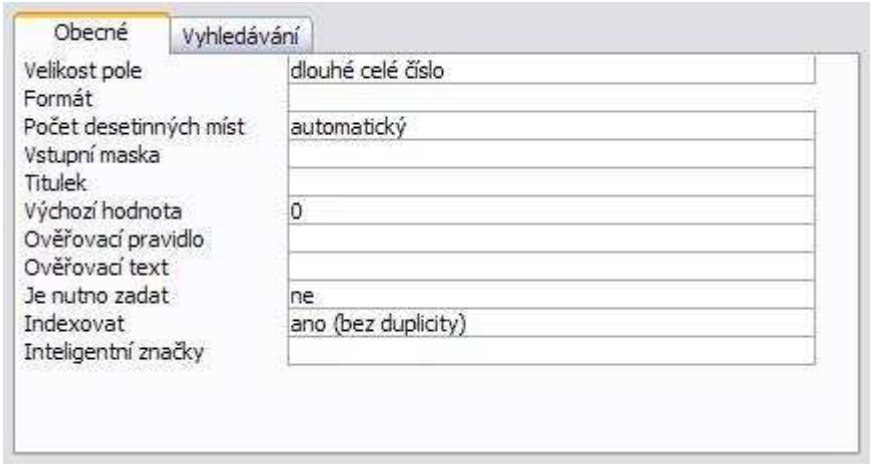
- Ano/Ne - položka typu Boolean - může obsahovat pouze dvě hodnoty, a to Ano (true, 1, On) a Ne (false, 0, Off). Velikost 1 bit.

- Objekt OLE - může obsahovat externí datové objekty (grafické soubory, textové dokumenty, tabulky, a jiná binární data) propojené, nebo vložené do tabulky programu MS Access. Velikost je omezena na 1 GB.

- Hypertextový odkaz – stejné jako typ Text, jen umožňuje pomocí hypertextového odkazu skočit na jiný soubor na disku nebo v síti.

3.1.3 Ostatní nastavení

V záložce Obecné si můžeme nastavit další vlastnosti pro pole.



Obr. 1 Vlastnosti pole pro datový typ číslo

Jak je vidět na obrázku Obr. 1, můžeme nastavit další vlastnosti, jako je formát zobrazení čísla, počet desetinných míst zobrazujících se v poli. Vstupní maskou řídíme způsob, jak uživatelé zapisují data do databáze. Vstupní masku je vhodné použít, pokud je třeba, aby uživatelé zadávali data určitým způsobem. Jestliže

například chceme, aby uživatelé zadávali telefonní čísla v britském nebo v německém formátu, použijeme vstupní masku.

Titulek je text, který se standardně zobrazí v popiscích ve formulářích, sestavách a dotazech. Výchozí hodnota, která se při přidávání záznamů automaticky přiřadí poli. Ověřovací pravidlo musí být při přidávání nebo změně hodnot v tomto poli pravdivé. Ověřovací text se zobrazí, jestliže zadaná hodnota nesplňuje ověřovací pravidlo. Vlastnost je nutno zadat, určuje zda je zadání dat do daného pole povinné. Vytvořením a použitím indexu můžeme zrychlit přístup k datům v tomto poli. Pomocí inteligentní značky připojíte k tomuto poli určitou značku. [4]

3.1.4 Relace

Při tvorbě tabulek konkrétní databáze bychom měli vzít v úvahu možné relace, které mohou vzniknout mezi tabulkami. Jedním z cílů kvalitního návrhu databáze je odstranit redundanci dat (duplicitní data). Dosáhneme toho tak, že data rozdělíme do mnoha tabulek s různými předměty, aby byl každý fakt reprezentován pouze jednou. Potom musím aplikaci Microsoft Access poskytnout prostředek, jak rozdělené informace opět spojit, a to umístěním společných polí do souvisejících tabulek, které budou mít definované vzájemné relace mezi sebou. [5]

Relace typu 1:1

V relaci 1:1 odpovídá jednomu záznamu v první tabulce maximálně jeden záznam v druhé tabulce a naopak jednomu záznamu v druhé tabulce maximálně jeden záznam v první tabulce. Tento typ relace není obvyklý, protože většina takto souvisejících informací bývá obvykle uložena ve stejné tabulce. Relaci 1:1 můžete použít k rozdělení rozsáhlé tabulky, k oddělení části tabulky z důvodů zabezpečení nebo k uložení informací, které mají vztah pouze k části hlavní tabulky. Při určování relace musí obě tabulky sdílet společné pole. [5]

Relace typu 1:N

Vezměme si jako příklad databázi pro sledování objednávek, jež obsahuje tabulky Zákazníci a Objednávky. Zákazník může vytvořit libovolný počet objednávek.

To znamená, že pro každého zákazníka uvedeného v tabulce Zákazníci, může existovat celá řada objednávek zaznamenaných v tabulce Objednávky. Typ relace mezi tabulkami Zákazníci a Objednávky je proto 1:N. [5]

Relace typu N:N

Chceme-li vyjádřit relaci typu N:N, musíme vytvořit třetí tabulku, která se často nazývá spojená tabulka, jež rozdělí relaci typu N:N na dvě relace typu 1:N. Primární klíče z těchto dvou tabulek vložíme do třetí tabulky. Výsledkem je, že třetí tabulka zaznamená každý výskyt nebo instanci relace. V relaci N:N jsou například tabulky Objednávky a Produkty a tato relace je definována vytvořením dvou relací 1:N s tabulkou Rozpis objednávek. V každé objednávce může být uvedeno více produktů a každý produkt může být uveden ve více objednávkách. [5]

Referenční integrita

Řekněme, že mezi tabulkami Přepravci a Objednávky máme relaci typu 1:N a chceme odstranit jednoho přepravce. Pokud pro tohoto přepravce existují objednávky v tabulce Objednávky, z těchto objednávek se po odstranění záznamu přepravce stanou osiřelé záznamy. Budou nadále obsahovat kód přepravce, ale ten nebude platný, protože záznam, na který odkazuje, už nebude existovat. [5]

Proto je vhodné nastavit v jednotlivých relacích také typ reference, aby nevznikaly osiřelé záznamy a nebo po aktualizaci záznamů v jedné tabulce byla i v dalších stále stejná a aktuální data.

3.2 Dotazy

Dotazy umožňují manipulovat s daty uloženými v databázových tabulkách. Můžeme je využívat k určení obsahu formulářů a sestav. Dotazy mohou být zdrojem pro webovou stránku. Dotazy umožňují data filtrovat, provádět s nimi výpočty a vytvářet jejich souhrny. Rovněž lze pomocí nich automatizovat mnoho úloh správy dat a kontrolovat změny dat před jejich potvrzením. [1]

Dotaz představuje požadavek na zobrazení výsledků dat, na provedení akce s daty nebo na kombinaci těchto dvou operací. Dotazy, které použijete k načtení dat z

tabulky nebo provedení výpočtů, se nazývají výběrové dotazy. Dotazy, které slouží k přidání, změně nebo odstranění dat, se nazývají akční dotazy. [1]

Pro tvorbu dotazů v aplikaci Access máme na výběr návrhové zobrazení, průvodce, a nebo složitější způsob pomocí jazyka SQL.

Výběrové dotazy

Vybírají data uložená v tabulkách a vrací výsledek ve tvaru datového listu, aniž by původní data změnily. Tento typ je vhodný zejména v případech , kdy chceme zobrazit pouze několik sloupců z mnoha sloupcové tabulky. Protože Access zpracovává tento dotaz při každém spuštění, uživatel tak získá vždy nejnovější údaje. [1]

Výběrový dotaz povoluje zpravidla aktualizovat pole v podkladových záznamech dotazu. Ve výběrových dotazech vytvořených z jedné nebo dvou tabulek spojených relací typu 1:1 jsou podkladové záznamy aktualizované vždy. Tabulky spojené relací 1:N jsou zpravidla také aktualizovatelné. [1]

To, zda je pole aktualizovatelné, zjistíme na stavovém řádku v zobrazení datového listu. V situaci, kdy pole nelze aktualizovat, avšak my tuto operaci přesto chceme realizovat, musíme otevřít tabulku přímo, a opravit hodnotu přímo tam. Jinou alternativou je změna struktury dotazu tak, aby pole bylo možné upravovat. [1]

Akční dotazy

Namísto vytváření výsledných sad kopírují nebo aktualizují data v jedné nebo několika tabulkách. Tyto dotazy zajišťují nepostradatelné funkce přidávání, odstraňování a aktualizování záznamů. Protože akční dotazy k vykonávání svých úkolů používají jazyk SQL, lze je považovat za sofistikované příkazy tabulek pro správu záznamů. [1]

Druhy akčních dotazů :

- Přidávací - přidávají záznamy na konec existující tabulky. Přidávací dotaz použijeme vždy, když bude aplikace vyžadovat záznamy pocházející z vnějšího zdroje. [1]

- Odstraňovací - odstraňují z jedné nebo z několika tabulek data jež vyhovují zadanému kritériu. Pomocí jednoho dotazu můžeme odstranit z tabulky dva i více záznamů. [1]

- Aktualizační - tento typ dotazu na rozdíl od ostatních nemění celé záznamy, ale koriguje hodnoty pouze ve vybraných polích. Zadáním kritéria nebo vyhledávací podmínky určíme příslušné řádky. [1]

Další typy dotazů

Tyto dotazy zpravidla, avšak ne vždy, rozšiřují výběrové a akční dotazy nebo tvoří jejich doplňky. [1]

- Parametrické – speciální typ, který může vracet řádky nebo vykonávat různé akce. Za běhu programu vyzve uživatele, aby zadal určitou hodnotu, jež bude řídit následující operaci. [1]

- Sjednocovací – kombinují odpovídající pole ze dvou nebo více tabulek nebo dotazů do jednoho pole. Podobá se to připojení tabulky k jiné tabulce. Tento dotaz je třeba sestavit pomocí jazyka SQL, nelze jej navrhnout v grafickém rozhraní návrhového zobrazení. [1]

- Křížové – počítají součet, průměr, počet nebo jiný typ souhrnu pole založeného na dvou dalších kategorických polích. Vyžaduje pole, které může aplikace použít pro své výpočty. [1]

- Poddotazy – je to příkaz SELECT jazyka SQL uvnitř jiného výběrového nebo akčního dotazu. Výsledná sada vnořeného příkazu SELECT se stane součástí vyhledávací podmínky hlavního dotazu. [1]

3.3 Formuláře

Všechny aplikace používají k prezentaci dat formuláře nebo sestavy. Formuláře mohou také přijímat uživatelský vstup anebo na něj reagovat. Protože uživatelé komunikují s databázovými aplikacemi téměř výhradně prostřednictvím formulářů, je jejich návrh důležitý. [1]

Úvodní formuláře :

Formuláře zobrazující se ihned po spuštění aplikace. Zobrazí se však ještě předtím, než se na obrazovce objeví první skutečně interaktivní formulář.

Úvodní formuláře zpravidla obsahují základní informace o účelu aplikace a o jejích tvůrcích. Délku zobrazení můžeme velmi snadno upravit pomocí sestavení události, kde zadáme čas v milisekundách.

Ústřední formuláře :

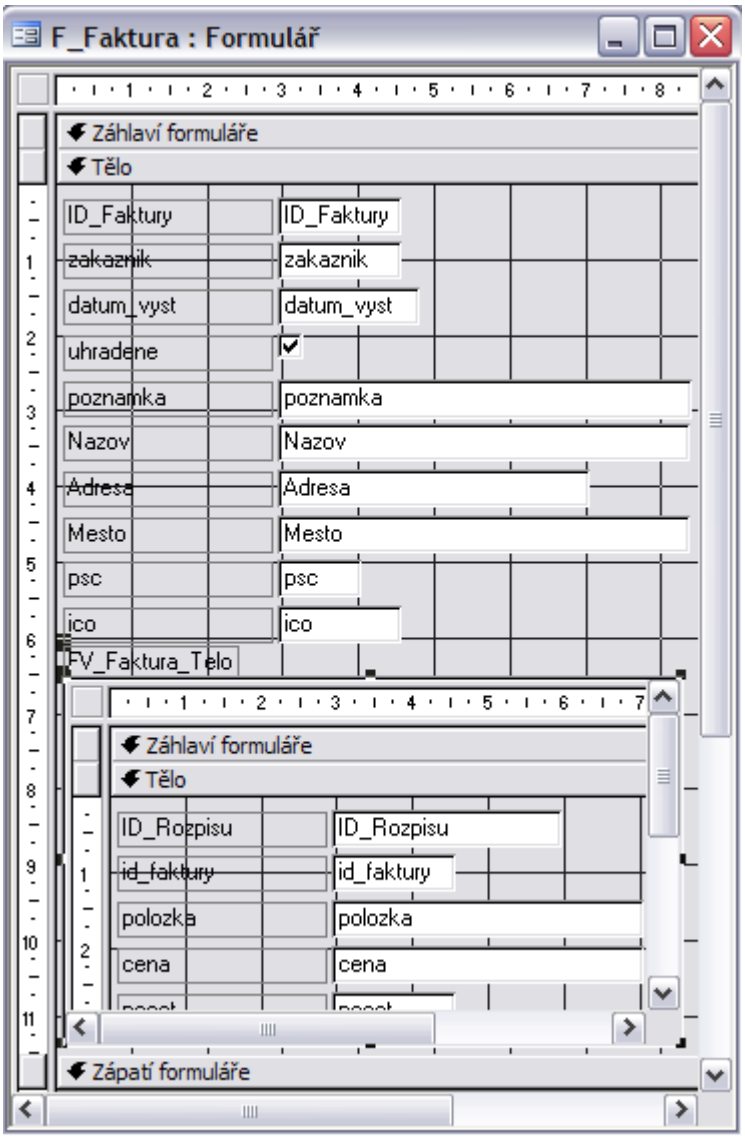
Většina formulářů obsahuje několik příkazových tlačítek, jejichž pomocí mohou uživatelé snadno otevřít příslušný formulář. V aplikaci můžeme využít navigaci pomocí navigačních tlačítek a nebo prostřednictvím programového kódu. [1]

Podformuláře :

Jedná se o způsob zobrazení dat ve formuláři, ve kterém je vložen podformulář. Hlavní formulář obsahuje obecné informace o objektu. Jeden nebo více hierarchicky souvisejících rozpisů se zobrazuje na jednom nebo několika podformulářích obsažených v hlavním formuláři.

Zdroj záznamů hlavního formuláře musí mít alespoň jedno společné pole se zdrojem záznamů každého svého podformuláře. Společné pole umožňuje, aby se v podformuláři zobrazily pouze údaje související s aktuálním záznamem v hlavním formuláři. Přesune – li se uživatel na jiný záznam v hlavním formuláři, podformulář bude obsahovat novou sadu záznamů, jež budou jedinečným způsobem spojeny s novým záznamem v hlavním formuláři. [1]

Na obrázku Obr. 2 vidíme názorný příklad vložení podformuláře do formuláře v návrhovém zobrazení. Jedná se o fakturu pro zákazníka, propojení obou formulářů je přes *ID_Faktury.* V hlavním formuláři jsou zobrazeny data související se zákazníkem, avšak v podformuláři je vypsaná položka, kterou si zákazník koupil na dané číslo faktury.



Obr. 2 Formulář s podformulářem

Vyhledávací formulář :

Nejjednodušší je v tomto případě nechat uživatele zadat příslušné informace do textového pole a stisknout tlačítko, které spustí vyhledávání. Textové pole by nemělo být vázané, protože jeho prostřednictvím nevkládáme do databáze žádné údaje. Toto pole jednoduše přijímá informace od uživatele. Dotaz pak využije tuto hodnotu k vyhledání shodné informace. [1]

Zobrazení výsledků může být ve formuláři, kde máme příslušné tlačítko k vyhledávání, a nebo dalším způsobem jako například okno se zprávou.

3.4 Sestavy

Ačkoliv sestavy nepodporují interaktivní ovládací prvky, můžeme je zaplnit ovládacími prvky zobrazujícími data jako například textovými poli nebo zaškrtávacími políčky. Sestavy mohou také obsahovat vázané i nevázané obrázky či ovládací prvky ActiveX pro speciální způsoby zobrazení dat jako například grafy. [5]

Při tvorbě sestavy je nejdůležitější určit zdroj záznamů sestavy. Ať je sestava jednoduchá, nebo se seskupenými záznamy je třeba nejprve určit pole, která obsahují data určená k zobrazení v sestavě, a tabulky nebo dotazy, v nichž se vyskytují.

Vytvářet sestavu můžeme několika způsoby. Buďto použijeme průvodce automatickou sestavou, další průvodce, a nebo si ji vytvoříme ručně v návrhovém zobrazení.

Průvodce automatickou sestavou :

Textová a číselná pole jsou zobrazena v textových polích, pole datových typů Ano/Ne jsou převedena na zaškrtávací políčka a pole obsahující binární data jsou obsažena v rámečcích vázaného objektu. Záznamy jsou zobrazeny jeden za druhým ve formátu datového toku, od prvního k poslednímu záznamu. [1]

Další průvodci :

Access poskytuje obecného průvodce sestavou, průvodce štítky a průvodce grafem. Průvodci mohou znatelně urychlit vývoj aplikace a přispět k její standardizaci, což ve svém důsledku usnadní její údržbu. Jsou dobrými pomocníky při návrhu vzhledu sestavy, usnadňují seskupování, řazení, či formátování sestav. [1]

Ruční vytváření v návrhovém zobrazení

Pokud vybereme tuto možnost návrhu, otevře se prázdný formulář, jehož základní vlastnosti odráží nastavení textového pole šablona sestavy. Základní šablonou je normální, která zobrazí prázdnou stránku bez žádných barev nebo speciálních typů písem pro štítky s adresami. [1]

3.4.1 Sekce sestavy

Návrh sestavy je v aplikaci Access rozdělen do sekcí. Chceme-li zobrazit sekce sestavy, můžeme ji zobrazit v návrhovém zobrazení. Chceme-li vytvářet užitečné sestavy, musíte vědět, jak jednotlivé sekce fungují. Například sekce, do které umístíme vypočítaný ovládací prvek, určí způsob výpočtu výsledků v aplikaci Access. Následující seznam je souhrnem typů sekcí a jejich použití. [5]

Záhlaví sestavy :

Tato sekce je vytištěna pouze jednou na začátku sestavy. Záhlaví sestavy se používá pro informace, které by se měly vyskytovat na titulní stránce, jako např. logo, nadpis nebo datum. Umístíme-li vypočítaný ovládací prvek, který používá agregační funkci *Sum*, do záhlaví sestavy, vypočítaný součet bude příslušet celé sestavě. Záhlaví sestavy je vytištěno před záhlavím stránky. [5]

Záhlaví stránky :

Tato sekce je vytištěna v horní části každé stránky. Záhlaví stránky lze použít například k uvedení názvu sestavy na každé stránce. [5]

Záhlaví skupiny :

Tato sekce je vytištěna na začátku každé nové skupiny záznamů. Záhlaví skupiny lze použít k tisku názvu skupiny. V sestavě, která je seskupena podle produktů, můžeme záhlaví skupiny použít například k tisku názvu produktu. Pokud do záhlaví skupiny umístím ovládací prvek, který používá agregační funkci Sum, vztahuje se součet na aktuální skupinu. [5]

Podrobnosti :

Tato sekce je vytištěna jednou pro každý řádek ve zdroji záznamů. Zde je třeba umístit ovládací prvky, které tvoří hlavní část sestavy. [5]

Zápatí skupiny :

Je vytištěno na konci každé skupiny. Zápatí stránky lze použít k tisku čísel stránek nebo informací zobrazovaných na každé stránce. [5]

Zápatí stránky :

Tato sekce je vytištěna na konci každé stránky. Zápatí slouží k tisku čísel stránek nebo informací určených pro každou stránku. [5]

Zápatí sestavy :

Tato sekce je vytištěna jednou na konci sestavy. Zápatí sestavy lze použít k tisku celkových součtů sestavy nebo jiných souhrnných informací za celou sestavu. [5]

3.5 Visual Basic for Applications

Programovací jazyk Visual Basic for Application (VBA), který je vestavěnou součástí databázové aplikace Access, poslouží především tam, kde jsou požadavky na výkon a ovládání vyšší, než které poskytují standardní nástroje aplikace Access. VBA se podobně jako makra používá k převodu databázových objektů do jednoho integrovaného celku. Správně naprogramovaná procedura v programovacím jazyku vykazuje s porovnáním s makry zásadně vyšší výkon. [1]

Vlastnosti a metody :

Charakterizují vzhled a chování objektů. Syntaxe, s jakou se na svoje objekty odkazují, je *objekt.vlastnost* nebo *objekt.metoda.* Termín objekt se může vztahovat jak na samotný objekt , tak na kolekci objektů.

Objekt *DoCmd* obsahuje mnoho metod , včetně metod *Close, OpenForm,GoToControl, FindRecord a RunCommand.* Řada metod *DoCmd*

přijímá argumenty, jež určují způsob provedení metody. Některé metody přijímají povinné i nepovinné argumenty. Pokud neurčím hodnotu nepovinného argumentu, metoda použije výchozí hodnotu. [1]

Mnoho metod objektu *DoCmd* se vztahuje přímo na jednotlivé objekty. Například metoda *GoToControl* přesune zaměření na konkrétní ovládací prvek na formuláři. Stejného výsledku dosáhneme použitím metody *SetFocus*, jež vybere ovládací prvek. Pokud chceme přesunout zaměření na určitý ovládací prvek, do něhož chceme vložit novou informaci nebo v kterém chcete opravit chybně zadaný údaj, můžeme snadno použít obě zmíněné metody. [1]

Události :

Události jsou jednou z nejdůležitějších součástí programování v jazyce VBA. Aplikace, která bude využívat události, bude dynamická a vůči uživateli interaktivní. Jak objekty tak kolekce obsahují události, jež slouží jako spouštěcí mechanismy pro jednotlivé části uživatelského programového kódu sestaveného vývojářem. Při práci s formulářem můžeme události používat k ověření správnosti vložených údajů, ke zpřístupnění nebo zakázání některých ovládacích prvků, ke změně zaměření či otevření nebo zavření formuláře. [1]

Ve vlastnostech formuláře máme na kartě *Událostní* výběr z mnoha událostí, které si hlídají formuláře samy, stačí jen zadat příslušný kód pro provedení akce. Tímto jsme zvýhodněni a nemusíme hlídat, při jakém příkazu se má provést definovaný požadavek.

3.6 Dokončení aplikace

V databázi se využívá hodně prvků, ať se jedná o formuláře nebo sestavy. Pro toho, kdo však nezná databázi, je však složité se zde orientovat. Proto je vhodné vytvořit vlastní ovládací nabídku a panel nástrojů pro každou databázi zvlášť, což zjednoduší práci většině uživatelů.

V záložce *Nástroje – Po spuštění* si nastavíme parametry, které nám budou vyhovovat při spuštění aplikace, například název databáze, ikonu databáze, nebo jaký formulář se zobrazí jako výchozí při zapnutí. Spousta dalších možností, které jdou nastavit je i možnost zobrazení nabídky kterou jsme si vytvořili ale také nezobrazovat původní.

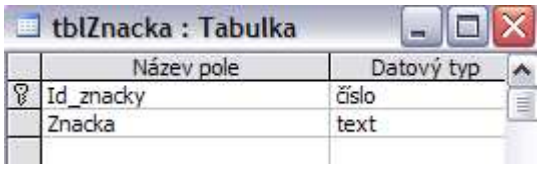
4. Návrh databázového programu

Pro majitele autoservisu bylo nevyhnutelné udělat tento krok a zavést databázi opravených vozidel, i když z počátku jim vyhovovalo jen vyřešit problém tisku faktur a jejich uchovávání pro pozdější reklamaci nebo informovanost o druhu opravy. Vzhledem k tomu, že papírové faktury by museli uchovávat ve své menší kanceláři v pořadačích nebo v policích, které by zabíraly místo ostatním věcem, rozhodl jsem se vyřešit tuto situaci právě databázovou aplikací s podporou požadovaných funkcí.

K doporučení konkrétního softwaru jsem neměl dostatek zkušeností s produkty tohoto typu, ale pokusil jsem se najít nějakou alternativu, která by vyhovovala požadavkům servisu. Výsledek byl však velice neuspokojivý, a proto jsem se rozhodl zhotovit aplikaci v rámci mé bakalářské práce. Požadavky byly jasně zadané, a to vyřešit tisk faktur a evidenci vozidel dle zadaných kritérií. Získané informace z technického průkazu sloužily k identifikaci vozidla a zjištění druhů náhradních dílu, pokud docházelo k výměně nebo opravě.

4.1 Popis a struktura tabulek

V tabulce *tblZnacky* jsou dva typy dat, číslo a text, přičemž každému číslu je přiřazena jedna značka vozidla. Pole číslo je v tabulce nastaveno jako primární klíč. Například pod číslem 1 se nachází značka Audi, a tak to jde až do čísla 33 kde jsem vypsal nejznámější značky vozů, které se nachází v evropských státech. Na obrázku Obr. 23 ji vidíme v návrhovém zobrazení.



Obr. 23 Tabulka Značky

Obr. 24 Tabulka Palivo

Tabulka *tblPalivo*, obrázek Obr. 24. Typově stejná jako tabulka *tblZnacky*, ale neobsahuje již žádné značky vozů, nýbrž druh paliva, jež se v dosavadní době používá pro pohon vozidel. Do těchto dvou zmiňovaných tabulek nebude uživatel moci zasahovat, byly naplněny daty dle požadavků majitele. Pokud se jedná o typ značky nebo druh paliva, tak servis je v této době omezen jen na typy mnou zadaných značek a druhů.

Obr. 3 Tabulka tblAuto

Obrázek Obr. 3 znázorňuje tabulku *tblAuto* v návrhovém zobrazení. Tabulka obsahuje primární klíč v poli *VIN,* které je typu číslo spolu s polem *RokVyroby* a *ID_Paliva*, ostatní jsou datového typu text.
ID_Znacka je cizím klíčem z tabulky *tblZnacky*, takže každé číslo, které je v tabulce může být i v poli druhé tabulky.

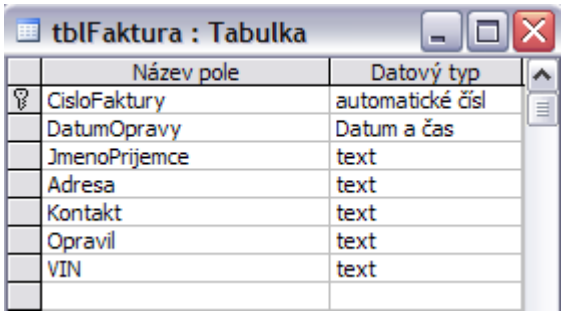
Do pole *Model* uživatel zadává model vybraného vozu odpovídající vybrané značce vozidla.

RokVyroby udává rok výroby automobilu. *ID_Paliva* je na stejném principu jako pole *ID_Znacka* jen je propojen s tabulkou *tblPalivo*.

VIN zadává uživatel z technického průkazu. Znamená mezinárodně jednoznačný identifikátor motorových vozidel, zpravidla vyražený na štítku trvale připevněném ke karoserii vozu nebo vyražený do karosérie samotné většinou na obtížně dostupné a záměnné části nosného skeletu. Ražení VIN se obvykle provádí předformovanými raznicemi až po lakování karosérie a v současné době bývá u modernějších vozů duplikováno i v jiných místech - typicky v průhledu stínícího lemu předního okna a v dalších, nezveřejňovaných místech.

V poli *Poznamky*, uživatel zaznamená další vlastnosti vozidla. Naříklad výkon motoru nebo typ karoserie. Velikost pole byla zvětšena z výchozí hodnoty 50 znaků na 120 znaků pro možnost připsání ostatních informací související s daným vozidlem.

SPZ znamená jednoznačné písmeno-číselné označení vozidla zaregistrovaného v určitém státu. Tabulka s tímto označením, nejčastěji ve formě bílé obdélníkové destičky s černými písmeny a čísly, je povinně umístěna na každém motorovém vozidle.



Obr. 4 Tabulka tblFaktura

Obrázek Obr. 4 znázorňuje tabulku *tblFaktura* v návrhovém zobrazení. Do tabulky se zadávají informace pro vedení faktur. Primárním klíčem v této tabulce je pole *CisloFaktury*, které je typu automatické číslo, aby uživatel nemusel zadávat pokaždé jiné číslo faktury, ale doplňovalo se samo a tím byl ušetřen čas.

DatumOpravy je typu Datum a čas a slouží pro přehled, kdy bylo vozidlo opraveno. Ostatní pole jsou datového typu text.

Adresa – adresa majitele vozidla je orientační, pokud by chtěli dopravit opravené vozidlo až domů a nebo pro přehled zákazníků využívajících jejich služeb.

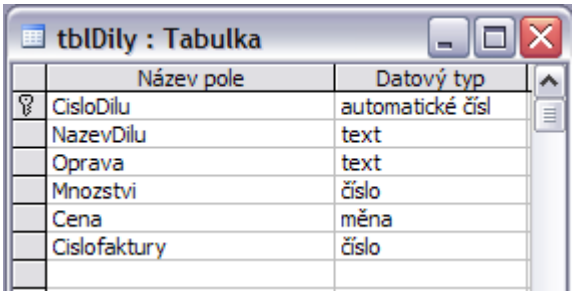
Kontakt - musí být vždy vyplněn, aby mohli technici ze servisu uvědomit majitele o dokončení opravy a kdy si může majitel pro svůj automobil přijet.

Opravil – udává, kdo automobil opravil, zaznamenává se pro přehled, který z mechaniků se podílel na opravě.

VIN – cizí klíč z tabulky *tblAuto*, zapisuje se aby bylo jasné pro který vůz je faktura vedena.

Obr. 5 Tabulka tblMajitel

Na obrázku Obr. 5 vidíme tabulku *tblMajitel* v návrhovém zobrazení. Tuto tabulku jsem zavedl kvůli vedení údajů o pracovních na dílně. Jelikož provádějí činnost na živnostenský list, každý z nich je jakoby majitel. Primárním klíčem tabulky je *Jmeno.* Tabulka obsahuje další pole *Příjmení, ICO, Adresa, Telefon, Telefon2,* kromě pole *ICO*, *Telefon* a *Telefon2* jsou všechny datového typu text.

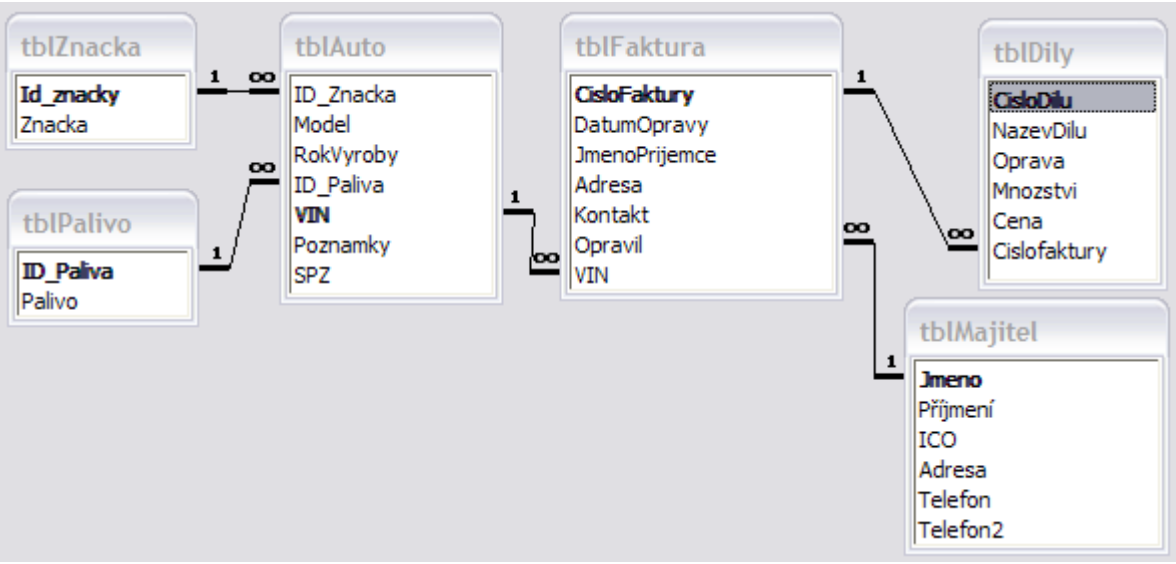


Obr. 6 Tabulka tblDily

Na Obrázku Obr. 6 je tabulka *tblDily.* Aby mohly být na faktuře uvedeny všechny druhy oprav pro dané číslo faktury, musel jsem přidat tuto tabulku, která obsahuje pole *CisloDilu* – primární klíč v tabulce typu automatické číslo, aby nemusel uživatel zadávat kolikátý díl byl opraven. *NazevDilu, Oprava* – typu text, udávají co bylo opraveno a jestli byl opravený díl vyměněn za nový, použitý a nebo druh práce vykonané na určitém automobilu. *Mnozstvi*, *CisloFaktury – typu číslo*, množství udává kolik dílu bylo opraveno.Číslo faktury je cizím klíčem z tabulky *tblFaktury*, kvůli přiřazení k faktuře.

4.2 Relace

Relace mezi tabulkami je znázorněna na obrázku Obr. 7. Mezi jednotlivými relacemi je zajištěna referenční integrita, aktualizace a odstranění souvisejících polí v kaskádě. Typ spojení mezi relacemi je všude stejný a to typ 1, který zahrnuje pouze řádky v nichž jsou spojená pole z obou tabulek shodná.

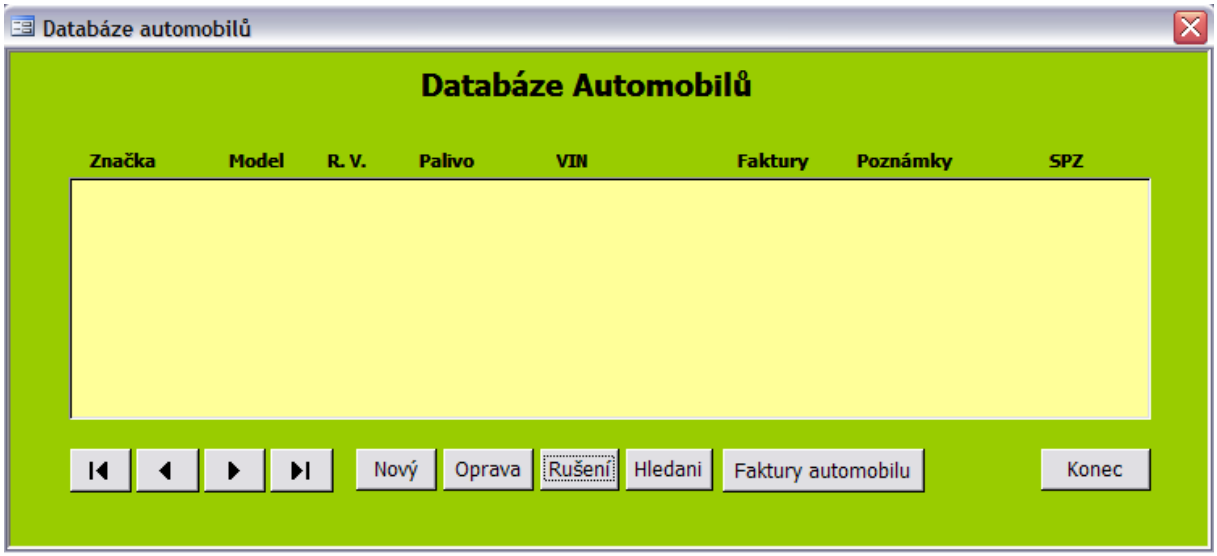


Obr. 7 Relace

4.3 Formuláře

V aplikaci jsem použil celkem 9 formulářů, které mají svoje ovládací a řídící tlačítka, tudíž není zapotřebí používat panel nabídek. Formuláře jsem vytvořil

v návrhovém zobrazení. Formuláře *Automobily*, *Faktury*, *Dily* jsou zarovnány na střed a nastaveny jako nepřesunutelné. Navigační tlačítka jako takové nebyly zapotřebí k navigaci mezi formuláři, v použitém seznamu by stejně nefungovaly jako navigace mezi záznamy, proto jsem je zakázal a vytvořil si vlastní tlačítka jen pro seznam. Posuvníky v aplikaci nevyužívám spolu s dalšími prvky, a to jsou voliče záznamů, dělící čáry, min a max tlačítka.



Obr. 8 Hlavní formulář

Na obrázku Obr. 8 vidíme hlavní formulář, s názvem Databáze automobilů, ve kterém jsou informace v seznamu. Pomocí tlačítek uživatel zadává, maže nebo hledá v seznamu informace o automobilu. Zdrojem formuláře je *DotazAuto*, ve kterém jsou všechny pole z tabulky *tblAuto*, *tblZnacka*, *tblPalivo* a také jedno pole z tabulky *tblFaktura* a to *CisloFaktury*, ve kterém je v souhrnná funkce Count ke zjištění počtu faktur pro automobil. Data jsou v dotazu řazena podle typu značky vzestupně. Mezi poli VIN z tabulky *tblAuto* a *tblFaktura* je nastaven typ spojení 2, jež zahrnuje všechny záznamy z tabulky tblAuto a z tabulky tblFaktura pouze ty záznamy, ve kterých jsou spojená pole shodná.

Tento hlavní formulář se zobrazuje po startu aplikace. Ve formuláři se můžeme mezi jednotlivými daty pohybovat pomocí navigačních tlačítek. Všechna tlačítka až na *Nový* a *Konec* jsou neaktivní, pokud nemáme žádné data v seznamu. K přidávání automobilů do seznamu slouží tlačítko *Nový,* toto tlačítko nevykonává

žádnou složitou proceduru, ale jen otevře formulář *NoveAuto.* Aby byly zadány všechny údaje, musel jsem zavést podmínku, která uživatele upozorní, pokud vynechal nějaké pole, a vrátí ho zpět k vyplnění do chybějícího pole. Není také možné zadat do pole jiný datový typ než je nastaven v tabulce. Pokud se tak stane, je uživatel upozorněn.

Značku a druh paliva uživatel vybírá z pole se seznamem používaných typů a značek. Pro pole VIN je nastaveno, že musí obsahovat vždy 17 znaků. Rok výroby má omezení, může se zadat rok výroby jen od roku 1960 do 2020. Toto časové rozmezí jsem zvolil proto, že se servis nezabývá opravou historických vozidel a také proto, aby mohli využívat tuto aplikaci i v budoucnu.

Po zadání všech dat se uloží záznam do tabulky tblAuto. Jestliže bude informace zadána chybně nebo bude neúplná, může jej kdykoliv změnit pomocí tlačítka *Oprava*. Tehdy se otevře formulář *OpravaAuto* s načtenými daty zvýrazněného automobilu v seznamu a mohou být změněny.

Ke smazání automobilu ze seznamu slouží tlačítko *Zrušit.* V tomto případě se vykoná příslušná událostní procedura, která je zobrazena na obrázku Obr. 9.

Než se automobil zruší, je nutné zjistit, jestli je v seznamu nějaký automobil. Pokud ano zjišťuje, se další podmínka nutná k odstranění, a to počet faktur, které automobil má. Když je počet faktur 0 může být smazán a provede se ještě dotaz na uživatele, jestli chce opravdu automobil smazat ze seznamu, a pokud je počet faktur větší než 0 vyzve to uživatele nejprve smazat faktury, aby mohl být automobil odstraněn se všemi daty uvedenými v evidenci. Tuto funkci jsem zavedl kvůli nechtěnému smazání automobilu, i když je uživatel ještě tázán k odstranění, tak jsem zavedl raději ještě jednu ochranu před smazáním důležitých dat. Po smazání se nedají už nijak vrátit data zpět.

Do proměnné **r** je zapsána aktuální pozice řádku automobilu, která po smazání slouží k tomu, aby se označil další řádek v seznamu.

```
If lstSeznam.ListCount > 0 Then
   If lstSeznam.Column(5) > 0 Then
      MsgBox "Automobil nelze zrušit, neboť má faktury.", vbInformation, "Upozornění"
   Else
      odp = MsgBox("Chcete zrušit dané auto?", vbQuestion + vbYesNo, "Rušení auta")
         If odp = vbYes Then
            r = lstSeznam.ListIndex
            Dim rstA As New ADODB.Recordset
            rstA.Open "tblAuto", CurrentProject.Connection, adOpenKeyset, adLockOptimistic
            rstA.Find "VIN='" & Forms.Automobily.lstSeznam & "'"
            rstA.Delete
            rstA.Close
            lstSeznam.Requery
               If r = lstSeznam.ListCount Then
                  r = r - 1
               End If
            lstSeznam = lstSeznam.ItemData(r)
         End If
   End If
```

Obr. 9 Událostní procedura tlačítka Zrušit

K odstranění bylo nutné zavést proměnnou RstA jako nový ADODB.Recordset, který vrací zdroj záznamů z tabulky *tblAuto* a napomáhá nám s nimi manipulovat. Najde nám automobil v tabulce podle VIN kódu, který je aktuální v sezamu, a smaže se jej i s ostatními daty v řádku nacházející se v tabulce s kódem VIN.

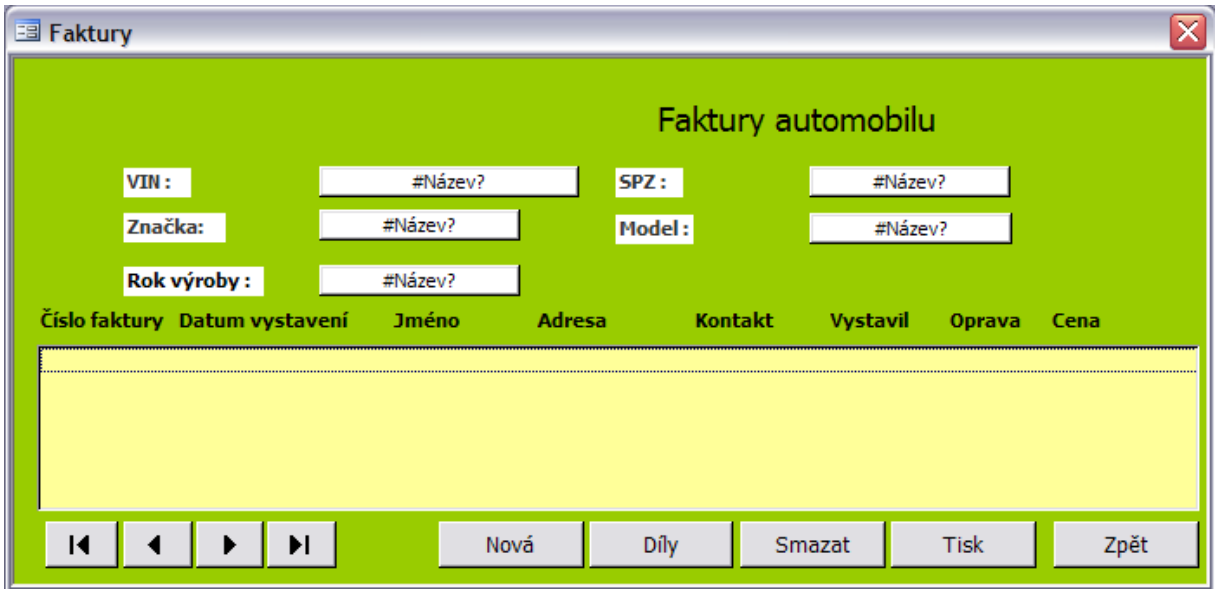
V naplněném seznamu uživateli napomáhá další tlačítko Hledat k nalezení automobilu podle dvou kritérií, VIN kódu a SPZ. Tlačítko otvírá formulář *Hledani.* Obrázek Obr. 10 znázorňuje kód z formuláře *Hledani* . V případě vyhledávání nemusí uživatel zadávat celý sedmnáctimístný kód VIN ani SPZ, protože do textového pole stačí zadat začáteční hodnoty, ty potom vyhledává jestli jsou nějaké podobné v seznamu.

Ve formuláři je textové pole, do kterého je nutné zadat hledaný výraz, jinak bude uživatel upozorněn, že nezadal žádnou informaci ke hledání. Pomocí dvou voleb si může vybrat mezi SPZ a VIN, pro upřesnění hledání je jako výchozí nastavena SPZ. Znovu jsem použil Recordset k nalezení záznamů v tabulce tblAuto.

```
Dim rstA As New ADODB.Recordset
rstA.Open "tblAuto", CurrentProject.Connection, adOpenKeyset, adLockOptimistic
If opgVolba = 1 Then
   rstA.Find " SPZ like '" & txtHledat & "*'"
Else
   rstA.Find "VIN like '" & txtHledat & "*'"
End If
If rstA.EOF Then
   MsgBox "vozidlo neexistuje !", vbInformation, "Upozornění"
   txtHledat.SetFocus
Else
   Forms.Automobily.lstSeznam = rstA!VIN
   DoCmd.Close
End If
rstA.Close
```

Obr. 10 Událostní procedura pro hledání

Když bude chtít uživatel přidat fakturu ke zvýrazněnému automobilu v seznamu, klikne na *Faktury automobilu*, který jej přesměruje na formulář *Faktury.* Můžeme jej vidět na obrázku Obr. 11



Obr. 11 Formulář faktury

Formulář *Faktury* načítá data ze seznamu z formuláře *Automobily,* a proto pokud jej spustíme samostatně, zobrazí se nám místo dat *#Název?.* Aby uživateli byla načtena data o automobilu, neumožnil jsem samostatné spuštění faktur uživatelem. Samostatný seznam má však zdroj záznamů dotaz *DotazF,* kde jsou všechna pole z tabulky *tblFaktura* a dvě pole z tabulky *tblDily* , jsou to *NázevDilu* a *Cena.*

Mezi poli NazevDilu z tabulky *tblFaktura* a *tblDily* je nastaven typ spojení 2, jež zahrnuje všechny záznamy z tabulky tblFaktura a z tabulky tblDily pouze ty záznamy, ve kterých jsou spojená pole shodná.

V dotazu je u pole *NazevDilu* jako nastavena souhrnná fukce Count, aby se zobrazovalo, kolik dílů je ve faktuře. Pole *Cena* má nastavenou souhrnnou funkci Sum, abychom věděli celkovou cenu všech dílů. V poslední řadě bylo důležité nastavit zobrazení faktury jen u automobilu, který jsme vybrali a to pomocí kritéria **[Forms].[Automobily].[lstSeznam].** Tímto nastaveným kritériem bylo zobrazení faktury vyřešeno.

V případě prázdného seznamu ve formuláři Faktury jsou aktivní pouze tlačítka *Nová* a *Zpět,* jelikož ostatní uživatel zatím nepotřebuje. Po načtení základních dat o automobilu z formuláře *Automobily* lze přidávat nové faktury pomocí tlačítka *Nová.*

To má obdobnou funkci jako tlačítko *Nový* v předchozím formuláři jen s tím rozdílem, že zadáváme jiná data a ukládají se do tabulky *tblFaktura.* Jako v předchozím formuláři má i tento formulář navigační tlačítka k pohybu mezi fakturami v seznamu.

Tlačítko *Smazat* smaže aktuální fakturu, i kdyby obsahovala nějaké díly. Jeho kód je podobný kódu, který je na obrázku Obr. 10. Pomocí Recordsetu najde se v tabulce číslo faktury a smaže celý řádek s daty pro vybranou fakturu. K vytisknutí faktury je zde tlačítko *Tisk,* které otevře sestavu s vyplněnými daty odpovídajícími vybrané faktuře.

Každá faktura by měla mít své díly, které byly opraveny a jelikož jich může být větší počet, slouží k tomu nový formulář *Dily,* jež se otvírá z formuláře *Faktury.* Jak je vidět z obrázku Obr. 12 tak i tento formulář potřebuje vstupní data z jiného formuláře a to *Faktury*, aby bylo jasné, ke které faktuře se mají díly přidat. Zdrojem dat pro seznam je dotaz *DotazDily* , jež obsahuje všechny pole z tabulky *tblDily* a jeho vstupní kritérium je **[Forms].[Faktury].[lstSeznam]**.

Obr. 12 Formulář Dily

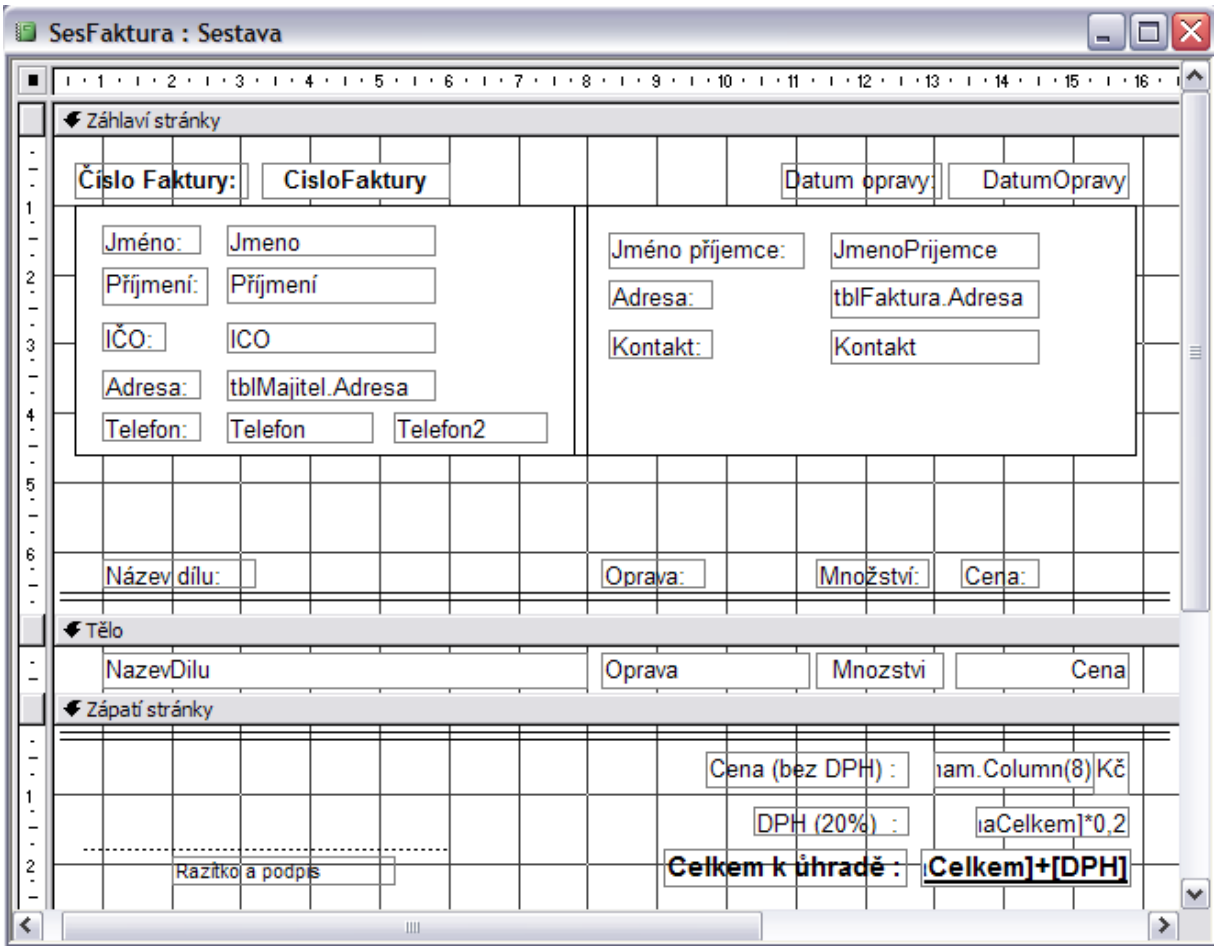
Pomocí tlačítek *Přidat*, *Opravit* a *Smazat* můžeme manipulovat s daty z tabulky *tblDily* načtené v seznamu. K pohybu mezi díly slouží navigační tlačítka.

4.4 Sestava

V této databázi využívám pouze jednu sestavu, a to k zobrazení faktury, za účelem tisku pro zákazníka. Sestava obsahuje údaje z dotazu DotazFaktura, v té jsou obsažena data z tabulek *tblFaktura*, *tblDily*, *tblMajitel* .

Struktura sestavy v návrhovém zobrazení je zobrazena na obrázku Obr. 13, kde vidíme jednotlivé uspořádání informací jak o podnikateli, tak o zákazníkovi a v neposlední řadě opravené součásti automobilu a jejich konečnou cenu včetně 20% DPH.

Pole kde se uvádí *cena*, *DPH*, a *celkem k úhradě,* se nachází v zápatí stránky, jsou vypočítány z údajů o ceně, která je v těle stránky. V sestavě se používá filtr **CisloFaktury=forms.faktury.lstSeznam**, aby se vytiskly jen díly obsažené ve vybrané faktuře.



Obr. 13 Sestava faktury pro tisk

4.5 Dokončení aplikace

V dialogovém okně Po spuštění jsem nastavil formulář *Automobily*, aby se spouštěl při startu aplikace, jelikož z něj vychází ostatní formuláře. Změnil jsem také původní název na *Evidence s fakturací* a přiřadil ikonu aplikace zobrazující se v hlavním okně, formuláři a sestavě. Výchozí nabídku a panel nástrojů nebude uživatel potřebovat, a proto jsem udělal novou nabídku se základními funkcemi, které uživatel využije. Panel nástrojů je uzpůsoben pouze pro tisk.

Jelikož se databáze zvětšuje s objemem obsahujících dat, nastavil jsem v možnostech komprimaci při zavření.

4.6 Zavedení databázové aplikace v servisu

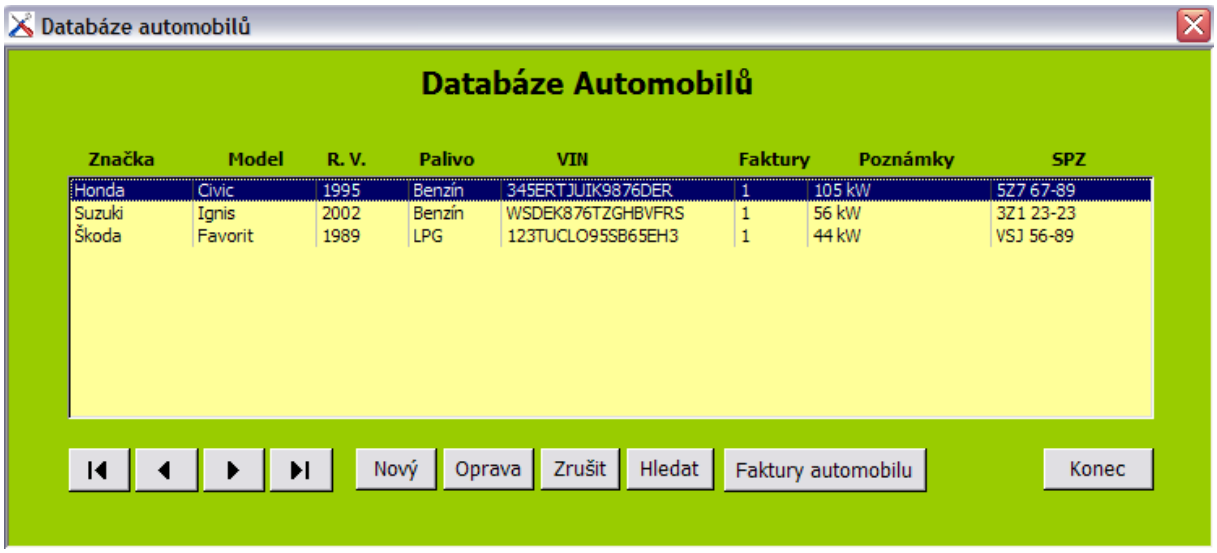
Na nový notebook, který zakoupil majitel v době mé odborné praxe, jsem nainstaloval sadu Microsoft Office 2003. Nebyl tedy problém se softwarem ani hardwarem, stačilo jen zkopírovat databázi na pevný disk. Kvůli bezpečnosti byl na notebooku rozdělen disk na dvě části, datovou a systémovou. Databázi jsem zkopíroval na část datovou. Pokud by spadl systém byly, by data z databáze nevyhnutelně ztracena. Spolu s databází je ve složce i obrázek typu ikona, odpovídající práci, kterou v servisu vykonávají. Tuto ikonu jsem nastavil zástupci na ploše, aby uživatel lépe poznal svou evidenční databázi.

Po prvním úspěšném spuštění databáze bylo nevyhnutelné seznámit její budoucí uživatele s její jednoduchou obsluhou. Přidávání automobilů, faktur a dílů je zcela jednoduché, hned po první ukázce si pracovníci tuto aplikaci vyzkoušeli a obstáli na výbornou.

4.7 Uživatelský návod

Ještě než začneme pracovat s aplikací, je nutné si říct pár základních informací o aplikaci. Slouží pouze pro evidenci automobilů a přidělování faktur, kde se popisují opravené závady. Všechna data, které obsahují, budou pouze ve vlastnictví auto-pneu servisu, pro nějž jsem tuto aplikaci navrhoval.

Pro spuštění aplikace je na ploše ikona s názvem *Evidence s fakturací,* dvojklikem na ní se spustí a zobrazí úvodní okno aplikace *Automobily,* jak vidíme na obrázku Obr. 14

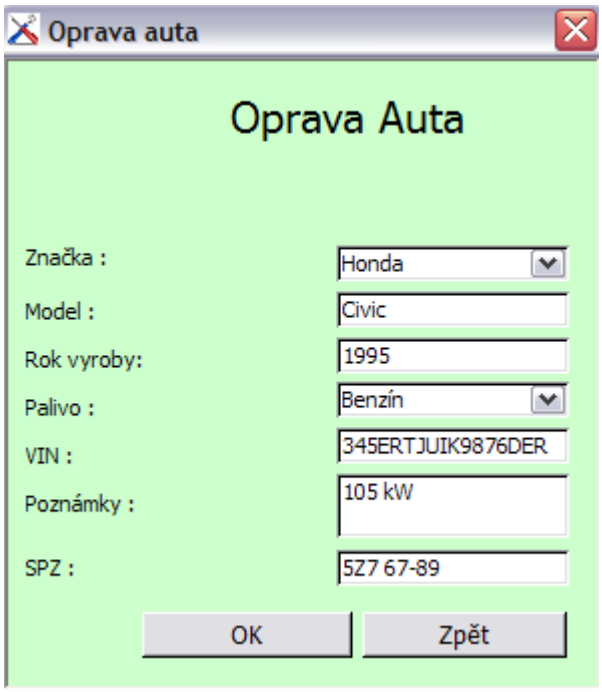


Obr. 14 Úvodní formulář se zkušebními daty

V tomto okně vidíme základní popis automobilu, pomocí navigačních tlačítek, které jsou dole vlevo, se posunujeme v seznamu. Tlačítko Nový otevře okno, kde přidáváme informace o automobilu do příslušných polí. Na obrázku Obr. 15 vidíme jednotlivé pole a způsob zadávání dat do seznamu. Značku automobilu a druh jeho paliva vybereme z možností uvedených v poli se seznamem. Ostatní informace zadáváme už podle druhu kam patří, například do pole, kde uvádíme rok výroby automobilu, můžeme zadat jenom číslice v rozmezí od 1960 do 2020.

Obr. 15 Přidávací okno pro automobily

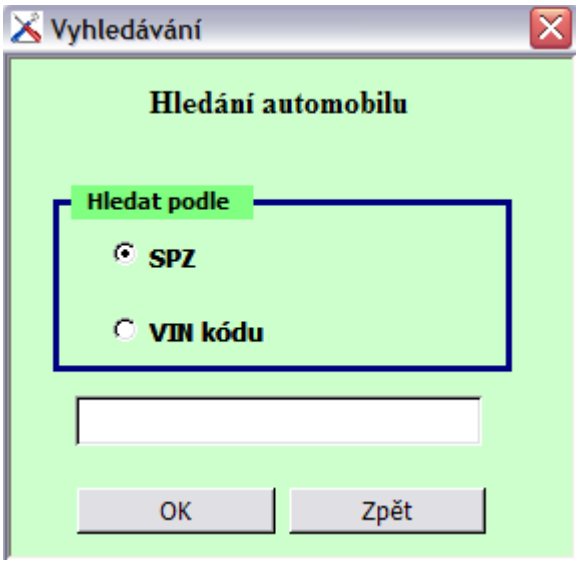
Pokud budou informace zadané špatně, nebo neúplné, slouží nám k tomu tlačítko opravit. Stačí si najet na příslušný automobil, a poté doplnit, čí změnit zadané údaje. Na obrázku Obr. 16 je zobrazeno okno pro opravy, s načtenými údaji před opravou.



Obr. 16 Oprava zadaných dat

Pokud se rozhodneme z nějakého důvodu odstranit automobil z evidence, například jestliže automobil již není nutno evidovat z důvodů sešrotování nebo nepojízdnému stavu, poslouží nám k tomu tlačítko *Zrušit.* Je zde nastaveno pouze jedno omezení, a to když má automobil nějaké faktury je nutné nejdříve odstranit faktury, aby nedocházelo k nechtěnému odstranění automobilu z evidence.

Ke snadnému nalezení automobilu v evidenci je zde tlačítko *Hledat.* Po kliknutí na něj se otevře okno pro zadání hledaného automobilu. Nabízí se nám dvě možnosti vyhledávání – podle VIN kódu automobilu nebo SPZ. Obrázek Obr. 17 znázorňuje, jak tohle okno vypadá. Výchozí volba je nastavena na SPZ, poté stačí do bílého pole zadat celou hodnotu SPZ nebo jen část, a tak je to i u VIN kódu. Není třeba vypisovat celý sedmnáctimístný kód ale postačí prvních pár znaků.



Obr. 17 Vyhledávání

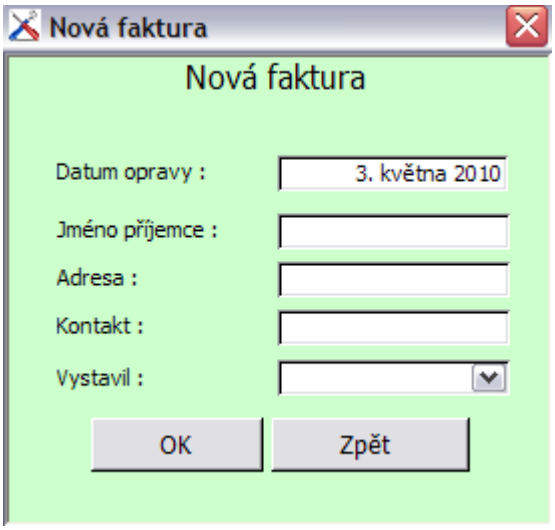
Tlačítko *Konec* zavře celou aplikaci a zkomprimuje celou databázi kvůli úspoře místa na disku, může se stát tedy že se aplikace zavře až po několika vteřinách.

K automobilu můžeme po zadání do seznamu ihned přidávat faktury, pomocí tlačítka Faktury automobilu se nám zobrazí v novém okně faktury jak můžeme vidět na obrázku Obr. 18 . V okně se nám zobrazí údaje o automobilu, který jsme vybrali. Na obrázku také vidíme již přidanou fakturu s jejími základními údaji.

Obr. 18 Faktury

Stejně jako hlavním okně i tady máme navigační tlačítka vlevo dole, sloužící k pohybu mezi záznamy v seznamu. K přidávání faktury zde máme tlačítko *Nová,* tím

otevřeme nové okno a zadáme příslušné údaje. Pro lepší pochopení jej vidíme na obrázku Obr. 19.



Obr. 19 Přidávání faktury

Datum opravy je automaticky generováno a nemusíme jej zadávat, což usnadní uživateli čas. Do ostatních polí vyplníme údaje týkající se majitele automobilu. Musím však podotknout, že Kontakt je brán pouze jako telefonní spojení na majitele. Pokud zadáme e-mail, icq a podobné komunikační formy, budeme upozorněni na zadání telefonního čísla. Naposled vyplníme kdo automobil opravoval.

Faktury můžeme kdykoliv smazat pomocí tlačítka *Smazat.* Není zde již žádné omezení jako v předešlém seznamu automobilů. Tlačítkem *Tisk* se zobrazí faktura v náhledu před tiskem, poté stačí v nabídce nástrojů vybrat ikonu s tiskárnou a faktura se odešle ihned do tiskové fronty. V příloze č.1 je zobrazena faktura se zkušebními daty. Tlačítkem *Zpět* se dostaneme zpátky na seznam automobilů.

Po úspěšném přidání faktury klikneme na tlačítko *Díly. Z*obrazí se nám okno s díly ve vybrané faktuře, jak je vidět z obrázku Obr. 20.

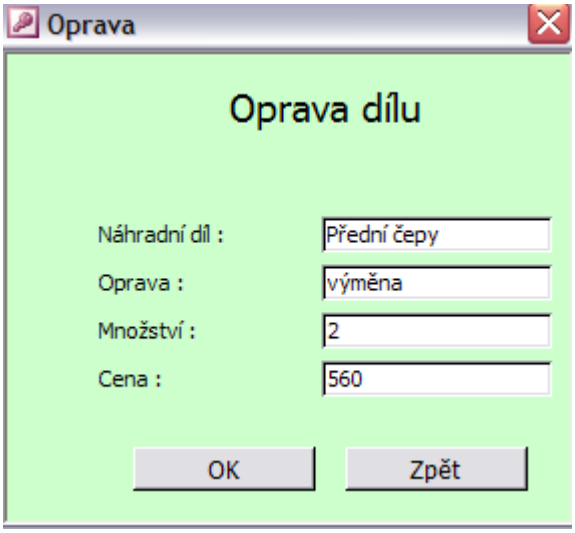
Obr. 20 Díly obsažené ve faktuře č.5

V okně s náhradními díly se zobrazí číslo faktury, pro kterou budeme přidávat díly. V seznamu máme informace o vyměněném dílu, druhu opravy, množství dílů a ceně. Tlačítkem Přidat otevřeme okno pro přidání dílu, jak je vidět na obrázku Obr. 21.

Obr. 21 Přidávací okno dílu

Cena a množství musí být zadána číselně, náhradní díl a opravu je také nutno vyplnit, jinak budeme navráceni zpět pro doplnění informací.

Jestliže se vyskytne v zadaném dílu chyba nebo špatné množství, využijeme tlačítko *Opravit.* S jeho pomocí se nám otevře nové okno s načtenými daty vybraného dílu, jak je vidět na obrázku Obr. 22.



Obr. 22 Oprava dílu

K mazání dílů slouží tlačítko *Smazat.* Mezi jednotlivými díly se také můžeme pohybovat pomocí navigačních tlačítek. Zpátky do okna s fakturami nás vrátí tlačítko *Zpět.* Po zadání všech dílu již můžeme tisknout kompletní fakturu.

5. Zhodnocení navrhovaného řešení

Doposud využívali v servisu pro tisk faktur vyplněný soubor s předešlými daty ve formátu Microsoft Excel. Soubor museli vždy přepisovat, pokud chtěli vytisknout další fakturu. Evidenční databáze s fakturací usnadní uživatelům hodně práce s přepisováním. Jelikož už budou informace o vozidlu vedeny v seznamu, stačí pouze přidávat faktury. Ještě k tomu budou mít data o automobilech vedena neustále v elektronické podobě, což bude velkým přínosem.

Díky aplikaci mohou technici zaznamenávat každou opravu automobilu, vystavit konkrétnímu automobilu fakturu a poté ji uchovávat v databázi, čímž dojde k úspoře místa, jež by zabíraly papírové faktury. Dále dochází k rychlejšímu vyhledávání informací o konkrétním automobilu a rychlejšímu zaznamenávání oprav a druhů vyměněných dílů. Tohle byl hlavní důvod pro vytvoření databáze. Vzhledem k zaznamenaným informacím se v servise vyvarují používání nekvalitních dílů, při další podobné závadě na automobilu.

K další výhodě mohu zařadit tisk faktur s automatickým výpočtem celkové ceny a DPH. Ze seznamu s díly se zjistí ceny všech opravených dílů a poté se dopočítá bez zásahu uživatele celková cena. Uživateli tak odpadá sčítání cen jednotlivých dílů a může si být jist, že jsou správně.

6. Závěr

Databázová aplikace, kterou jsem popisoval v bakalářské práci, byla vytvořena pro auto-pneu servis Lukáše Burdíka. Podle jeho kritérií jsem narhoval tabulky pro budoucí data vhodná k dlouhodobé evidenci. Vzhled a ovládání jsem navrhoval samostatně, a to spíše podle využívaných programů v servisu. Barevné schéma jsem zvolil skoro ve všech formulářích v odstínech zelené a žluté barvy, aby na uživatele nepůsobila aplikace příliš chladně.

Druhá kapitola obsahuje stručnou charakteristiku auto-pneu servisu s popisem nabízených služeb a uvádí hardware a software využívaný v podnikání. V třetí kapitole se seznámíme se základními prvky aplikace Microsoft Access a programovacího jazyka Visual Basic for Applications. Ve čtvrté kapitole popisuji již navrhovanou databázi. Popisuji jednotlivé faktury, vazby mezi tabulkami, formuláře a sestavy. V Páté kapitole hodnotím dosažené výsledky při zkušebním zavedení aplikace.

Mým hlavním cílem bylo vytvořit aplikaci pro evidenci automobilů a faktur k nim přiřazených, což se mi povedlo zrealizovat. Aplikace je celkem jednoduchá na používání, k jakékoliv změně jsou v každém okně tlačítka pro příslušnou manipulaci s daty. Dodržel jsem všechny zásady pro přidávání dat do seznamu, omezení o délce, hodnotě a povinnosti zadávat data. Grafické rozhranní vyhovuje lépe než předešlý využívaný soubor v Excelu.

V závěru lze konstatovat, že zmiňovaná aplikace vyhovuje ve všech směrech zadané problematice, pro kterou byla vytvořena.

Seznam použité literatury

Knihy :

1. DOBSON, Rick. *Programování v Microsoft Access 2000*. 1. vyd. Brno : Computer Press, 2000. 542 s. ISBN 80-7226-271-8.

2. NOEL , Jerke. *Microsoft Office Access 2003 pro pokročilé*. 1. vyd. Brno : Computer press, 2005. 352 s. ISBN 80-251-0723-X.

3. VIESCAS, John L. *Mistrovství v Microsoft Office Access 2003* . 1. vyd. Brno : Computer Press, 2005. 968 s. ISBN 80-251-0537-7.

Elektronické zdroje :

4. KOCAN, Marek. *Www.dbsvet.cz* [online]. 15. 10. 2007 [cit. 2010-04-27]. Informační portál ze světa databázových technologií. Dostupné z WWW: http://www.dbsvet.cz/view.php?cisloclanku=2007101501**.**

5. *Www.office.microsoft.com* [online]. 2007 [cit. 2010-04-27]. Dostupné z WWW: http://office.microsoft.com.

Seznam zkratek

IT - Informační technologie

STK - Státní technická kontrola

SQL - Structured Query Language (strukturovaný dotazovací jazyk)

VBA - Visual Basic for Applications (programovací jazyk)

VIN - Vehicle identification number (Identifikační číslo vozidla)

SPZ - Státní poznávací značka

IČO - Identifikační číslo ekonomického subjektu

Seznam příloh

Příloha č.1 : Sestava faktury (se zkušebními daty)

Příloha č.2 : CD – obsahuje vytvořenou aplikaci

Příloha č.1 (další strana)

Číslo Faktury: 5 Datum opravy 3. května 2010

Jméno:	Lukáš		Jméno příjemce:	Miroslav Maňák
Příjmení:	Burdík		Adresa:	Huslenky černé 87
IČO:	87228769		Kontakt:	737145078
Adresa:	Vsetín			
Telefon:	732873286	736786453		

Název dílu:	Oprava:	Množství:	Cena:
Brzdové obložení	výměna	2	1 200 Kč
Přední čepy	výměna	2	560 Kč

Cena (bez DPH) :	1760 Kč
DPH (20%) :	352 Kč
Celkem k úhradě :	**2 112 Kč**

.....
Razítko a podpis